500P0430**00

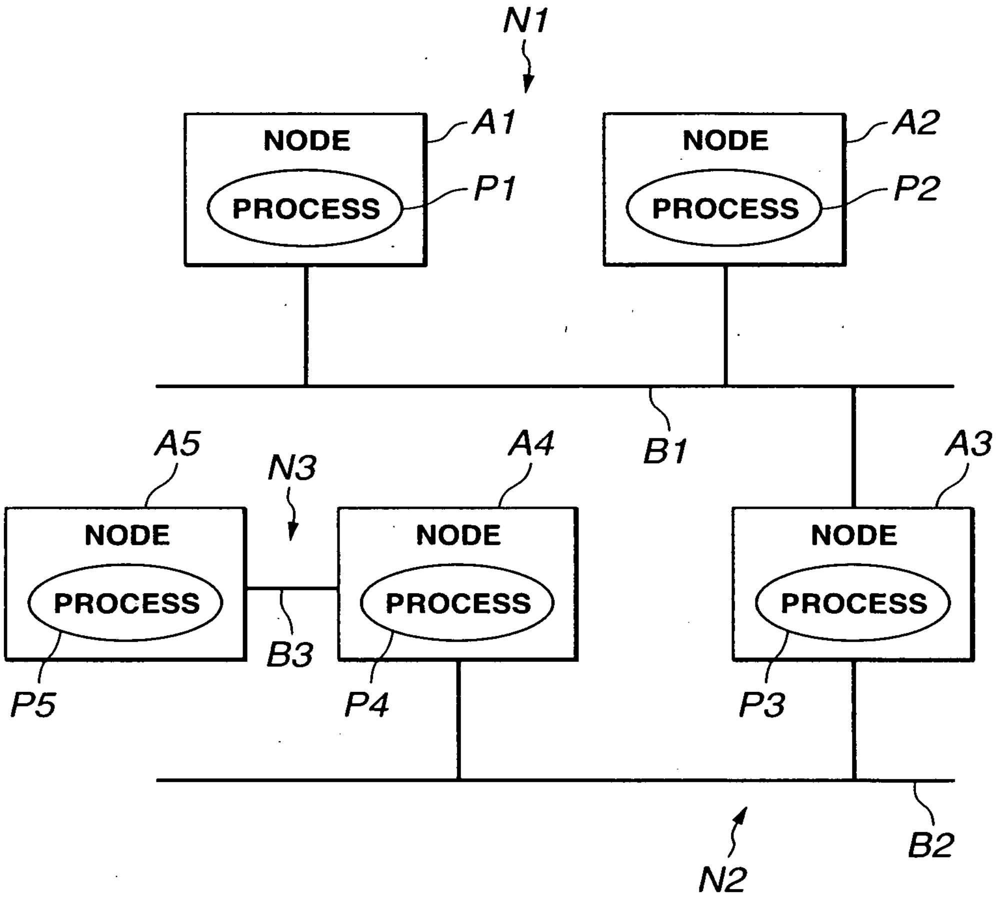

FIG.1

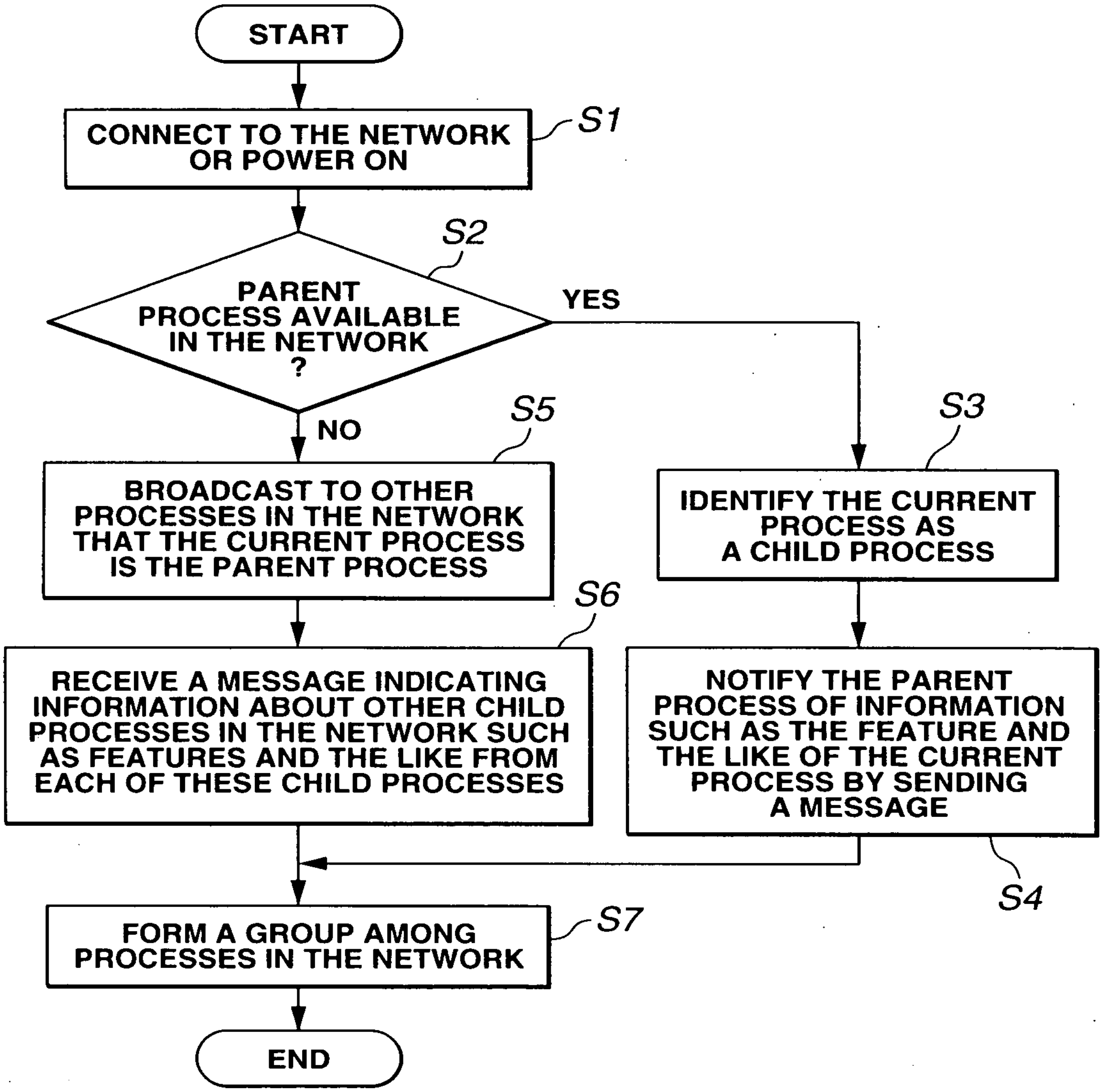
START
CONNECT TO THE NETWORK OR POWER ON
S1
PARENT PROCESS AVAILABLE IN THE NETWORK ?
S2
YES
NO
BROADCAST TO OTHER PROCESSES IN THE NETWORK THAT THE CURRENT PROCESS IS THE PARENT PROCESS
S5
IDENTIFY THE CURRENT PROCESS AS A CHILD PROCESS
S3
RECEIVE A MESSAGE INDICATING INFORMATION ABOUT OTHER CHILD PROCESSES IN THE NETWORK SUCH AS FEATURES AND THE LIKE FROM EACH OF THESE CHILD PROCESSES
S6
NOTIFY THE PARENT PROCESS OF INFORMATION SUCH AS THE FEATURE AND THE LIKE OF THE CURRENT PROCESS BY SENDING A MESSAGE
S4
FORM A GROUP AMONG PROCESSES IN THE NETWORK
S7
END

FIG.2

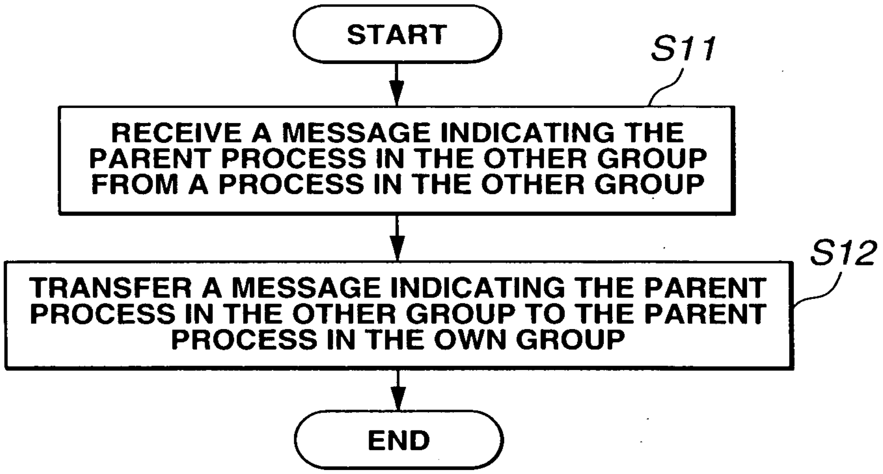
START
S11
RECEIVE A MESSAGE INDICATING THE PARENT PROCESS IN THE OTHER GROUP FROM A PROCESS IN THE OTHER GROUP
S12
TRANSFER A MESSAGE INDICATING THE PARENT PROCESS IN THE OTHER GROUP TO THE PARENT PROCESS IN THE OWN GROUP
END

FIG.3

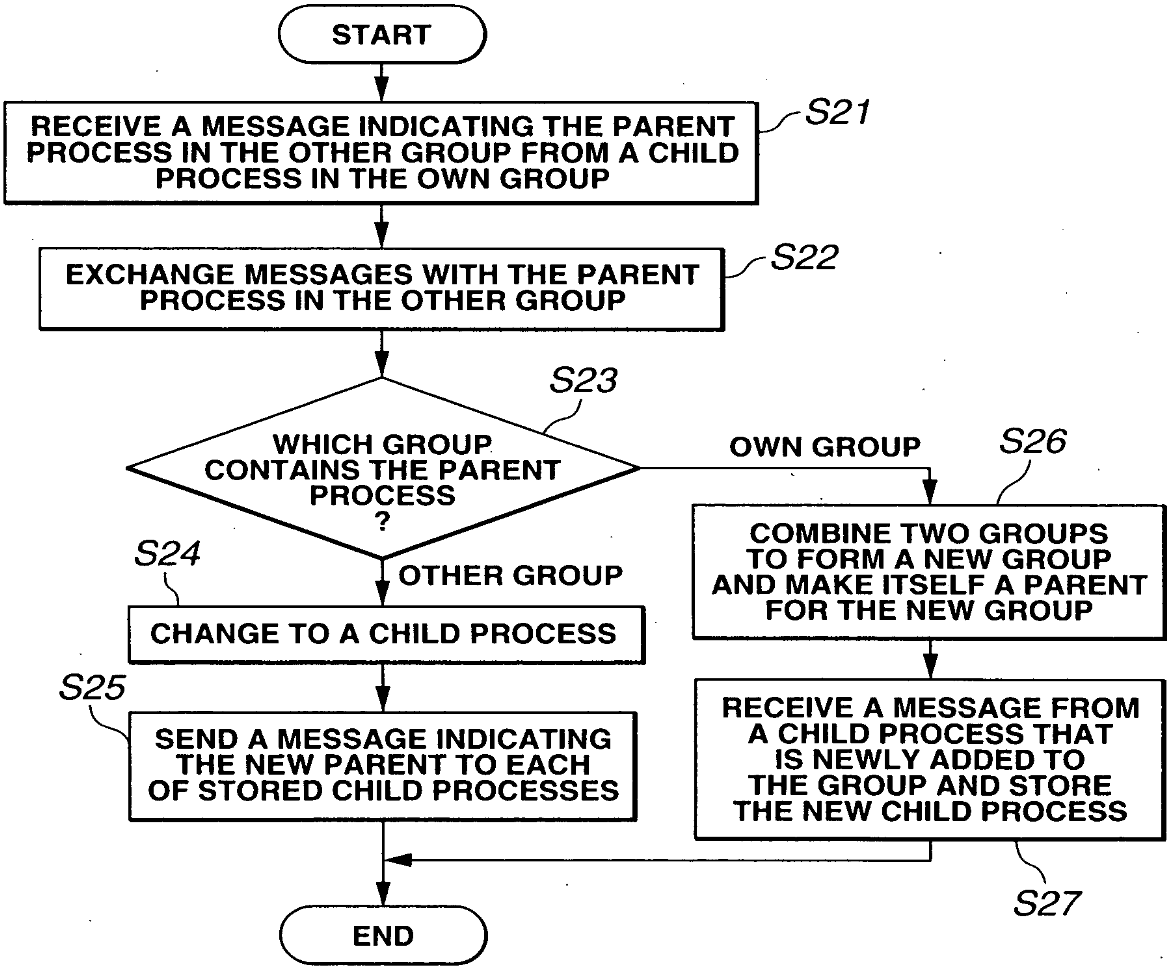
START
S21
RECEIVE A MESSAGE INDICATING THE PARENT PROCESS IN THE OTHER GROUP FROM A CHILD PROCESS IN THE OWN GROUP
S22
EXCHANGE MESSAGES WITH THE PARENT PROCESS IN THE OTHER GROUP
S23
WHICH GROUP CONTAINS THE PARENT PROCESS ?
OWN GROUP
S26
OTHER GROUP
S24
CHANGE TO A CHILD PROCESS
S25
SEND A MESSAGE INDICATING THE NEW PARENT TO EACH OF STORED CHILD PROCESSES
COMBINE TWO GROUPS TO FORM A NEW GROUP AND MAKE ITSELF A PARENT FOR THE NEW GROUP
RECEIVE A MESSAGE FROM A CHILD PROCESS THAT IS NEWLY ADDED TO THE GROUP AND STORE THE NEW CHILD PROCESS
S27
END

FIG.4

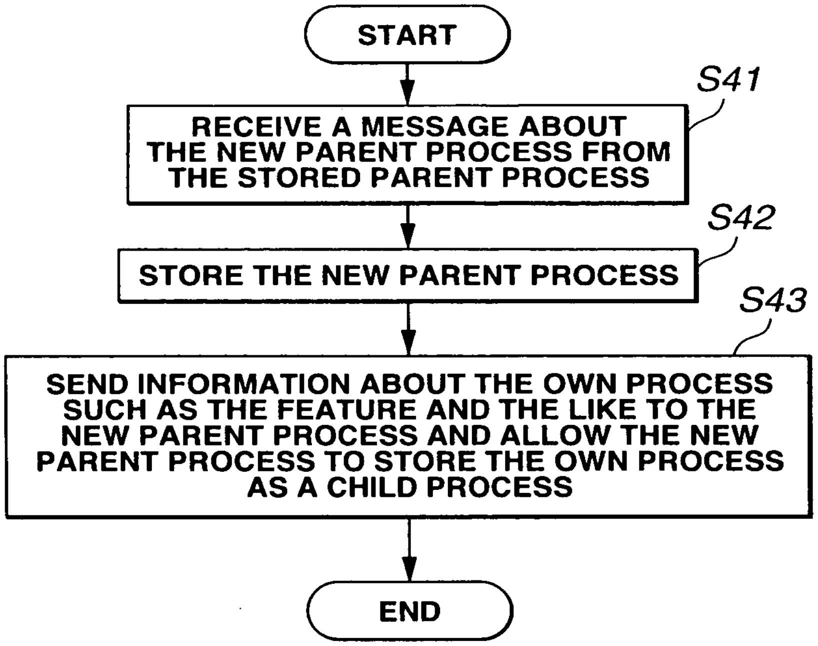
START
S41
RECEIVE A MESSAGE ABOUT THE NEW PARENT PROCESS FROM THE STORED PARENT PROCESS
S42
STORE THE NEW PARENT PROCESS
S43
SEND INFORMATION ABOUT THE OWN PROCESS SUCH AS THE FEATURE AND THE LIKE TO THE NEW PARENT PROCESS AND ALLOW THE NEW PARENT PROCESS TO STORE THE OWN PROCESS AS A CHILD PROCESS
END

FIG.5

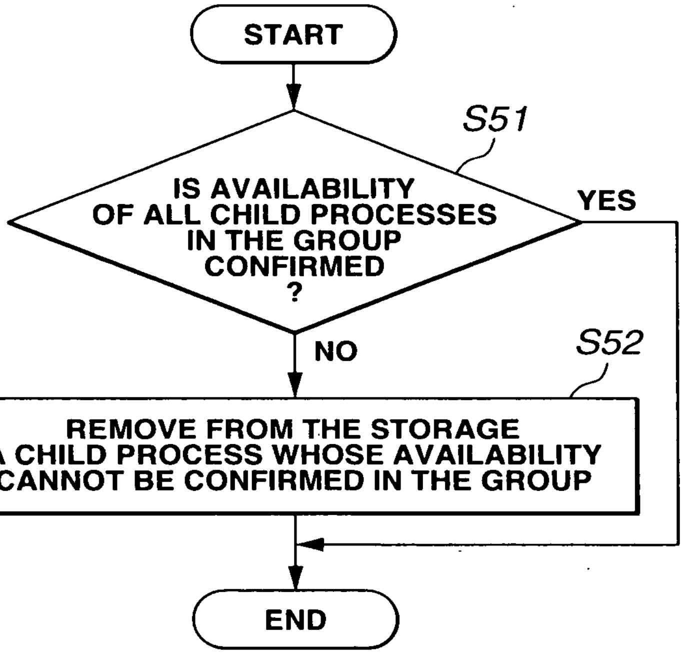
START
S51
IS AVAILABILITY OF ALL CHILD PROCESSES IN THE GROUP CONFIRMED ?
YES
NO
S52
REMOVE FROM THE STORAGE A CHILD PROCESS WHOSE AVAILABILITY CANNOT BE CONFIRMED IN THE GROUP
END

FIG.6

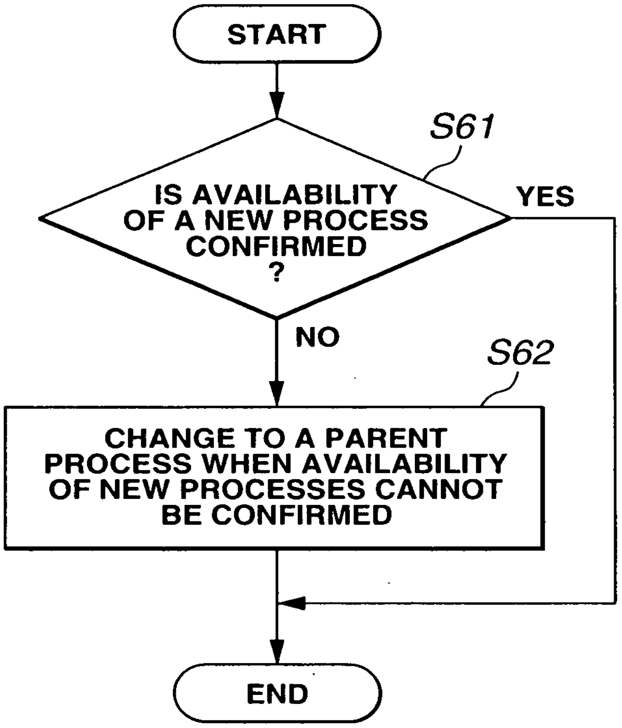
START
S61
IS AVAILABILITY OF A NEW PROCESS CONFIRMED ?
YES
NO
S62
CHANGE TO A PARENT PROCESS WHEN AVAILABILITY OF NEW PROCESSES CANNOT BE CONFIRMED
END

FIG.7

```
loop{
  wait

      if (S==S_PARENT) {
        if (E . m_type==M_NOTIFY) {
           if (E . m_addr !=M) (
              assign_P (E. m_addr) ;
              send (P, M_NEGOTIATE , M) ;
              trans (S_NEGOTIATING) ;
           }
         }else
         if (E . m_type==M_FORWARD) {
           ignore ( ) ;
         }else
         if (E . m_type==M_REPORT) {
           if (E . m_addr !=M) {
              assign_P (E. m_addr) ;
              send (P, M_NEGOTIATE , M) ;
              trans (S_NEGOTIATING) ;
           }
         }else
         if (E . m_type==M_NEGOTIATE) {
           assign_P (decide (E. m_addr , M) ;
           if (P==M) {
              send (E . m_addr , M_IAM ,M)) ;
              trans (S_NEGOTIATING) ;
     } else {
           send (E . m_addr , M_YOUARE) ;
      send (P. M_JOIN , M) ;
      send_C (M_FORWARD , P) ;
      empty_C ( ) ;
      trans (S_CHILD) ;
      }
        }else
        if (E . m_type==M_IAM) {
      ignore ( ) ;
         }else
         if (E . m_type==M_YOUARE) {
      ignore ( ) ;
        }else
        if (E . m_type==M_BUSY) {
          ignore ( ) ;
        }else
        if (E . m_type==M_JOIN) {
          add_C (E . m_addr) ;
        }else
        if (E . m_type==M_ERROR) {
          ignore ( ) ;
        }else
        if (E . m_type==M_TIMOUT) {
          broadcast (M_NOTIFY , M) ;
        }
      }else
```

FIG.8

```
if (S==S_NEGOTIATING) {
    if (E . m_type==M_NOTIFY) {
        ignore ( ) ;
    }else
    if (E . m_type==M_FORWARD) {
 if (E . m_addr==M) {
       assign_P (M) ;
       trans (S_PARENT) ;
 }else{
          assign_P (E. m_addr) ;
          send (P , M_NEGOTIATE , M) ;
 }
    }else
    if (E . m_type==M_REPORT) {
       ignore ( ) ;
    }else
    if (E . m_type==M_NEGOTIATE) {
 if (P==E . m_addr && decide (P , M)==P) {
    send (E . m_addr , M_YOUARE) ;
 }else{
   send (E . m_addr , M_BUSY) ;
 }
     }else
     if (E . m_type==M_IAM) {
        assign_P (E . m_addr) ;
        send (P , M_JOIN , M) ;
        send_C (M_FORWARD , P) ;
        empty_C ( ) ;
        trans (S_CHILD) ;
     }else
     if (E . m_type==M_YOUARE) {
        assign_P (M) ;
        trans (S_PARENT) ;
     }else
     if (E . m_type==M_BUSY) {
        send (P , M_NEGOTIATE , M) ;
     }else
     if (E . m_type==M_JOIN) {
        add_C (E . m_addr) ;
     }else
     if (E . m_type==E_ERROR) {
        assign_P (M) ;
        trans (S_PARENT) ;
     }else
     if (E . m_type==E_TIMEOUT) {
        assign_P (M) ;
        trans (S_PARENT) ;
     }
 }else
```

FIG.9

```
   if (S==S_CHILD) {
      if (E . m_type==E_NOTIFY) {
         if (E . m_addr  !=P) {
           send (P , M_REPORT , E . m_addr) ;
         }
      }else
      if (E . m_type==E_FORWARD) {
        assign_P (E . m_addr) ;
    if (P==M) {
      trans (S_PARENT) ;
    }else{
     send (P , M_JOIN , M) ;
    }
      }
      if (E . m_type==M_REPORT) {
        send (E . m_form , M_FORWARD , P) ;
      }else
      if (E . m_addr , M_FORWARD , P) ;
        send (E . m_form , M_FORWARD , P) ;
      }else
      if (E . m_type==M_IAM) {
        ignore ( ) ;
      }else
      if (E . m_type==M_YOUARE) {
        ignore ( ) ;
      }else
      if (E . m_type==M_BUSY) {
        send (P , M_JOIN , M) ;
      }else
      if (E . m_type==M_JOIN) {
        send (E . m_addr , M_FORWARD , P) ;
      }else
      if (E . m_type==M_ERROR) {
        assign_P (M) ;
        trans (S_PARENT) ;
      }else
      if (E . m_type==E_TIMEOUT) {
        broadcast (M_NOTIFY , P) ;
        trans (S_PARENT) ;
      }
    }
  }
}
```

FIG.10

```
addr <information> D[]
addr <information> D[]
addr <information> D[]
          :
```

FIG.11

SERVICE IDENTIFIER	ADDRESS	PORT NUMBER
⋮	⋮	⋮

FIG.13

```
loop{
  Update own information as required
  Delete all elements from D[] for the updated line and add M to it

   if (M_COPY message arrived)
      Modify the table according to the message
      Delete all elements from D[] for the updated line and add addr to it
   }
   if (M_DELETE message arrived) {
      Delete a line with matching addr from the table
   }
   if (this process is parent) {
     for (all lines in the table) {
       if (deleting a line whose addr is not contained in C[]) {
         for (all elements T in C[]) {
            send (T , M_DELETE , addr)
         }
       }
     }
     for(all lines in the table) {
       for (C[] elements T that are not contained in D[]) {
         send (T , M_COPY , addr , <information>)
         Add T to D[]
         }
       }
    }  else {
       for (all lines in the table)
         if (addr==M and P not contained in D[])
            send (P , M_COPY , M , <information>)
            Add P to D[]
         }
       }
     }
   }
```

FIG.12

SERVICE IDENTIFIER	ADDRESS	PORT NUMBER
WWW-service	192 . 168 . 1 . 1	80
file-service	192 . 168 . 1 . 1	2049
print-service	192 . 168 . 1 . 1	515

FIG.14

addr	INFORMATION			D[]
	SERVICE IDENTIFIER	ADDRESS	PORT NUMBER	
192 . 168 . 1 . 1	WWW-service	192 . 168 . 1 . 1	80	···
	file-service	192 . 168 . 1 . 1	2049	
	print-service	192 . 168 . 1 . 1	515	
192 . 168 . 1 . 2	WWW-service	192 . 168 . 1 . 2	80	···
	print-service	192 . 168 . 1 . 2	515	
192 . 168 . 1 . 3	WWW-service	192 . 168 . 1 . 3	80	···
	file-service	192 . 168 . 1 . 3	2049	

FIG.15